Отчёт по производственной практике.

Пестов Е.Ю. ИСПо332

Тема: 6. Работа с Jira. Создание баг-репорта.

1. **Что такое** **Jira? Для чего используют Jira.**

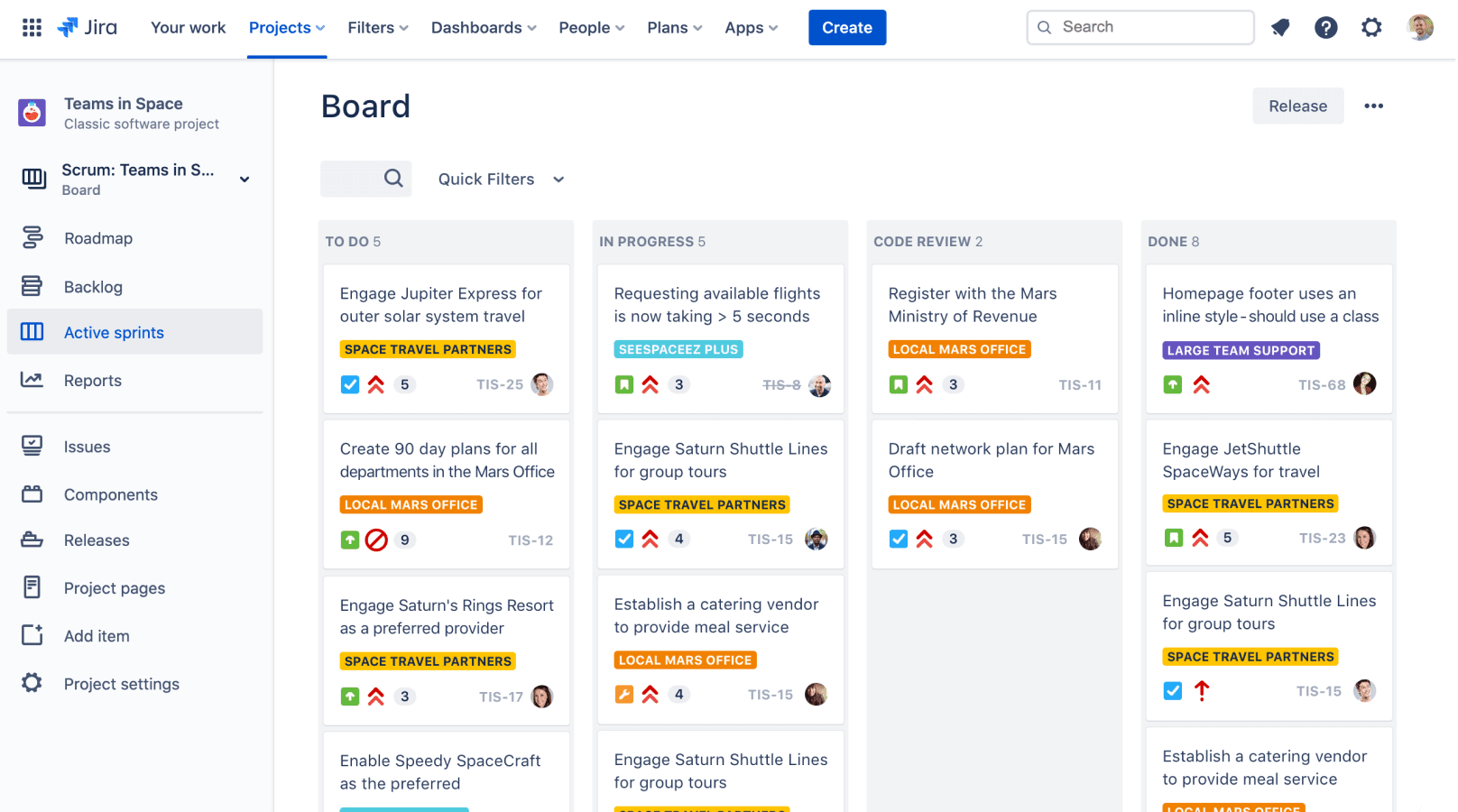
Jira – это программный инструмент для управления проектами, разработанный компанией Atlassian. Jira часто используется в IT-компаниях для формирования списка задач, отслеживания общего прогресса команды и решения возникающих по ходу разработки продукта проблем. Приложение Atlassian построено по принципам канбан- и скрам-досок, давней практики организации задач. Но эти принципы дополняются массой вспомогательных механизмов, которые добавлялись в приложение исключительно с целью упростить создание новых приложений, добавить в них функции, исправить ошибки и т.п. Также эта система управления проектами использует Agile-методику разработки. Название происходит от японского слова Gojira, что переводится как «Годзилла». [1]

С помощью Jira команды могут планировать, назначать и отслеживать работу, а также управлять ею и создавать отчеты. Jira позволяет объединяться для любой работы, от agile-разработки ПО и поддержки клиентов до организации стартапов и крупных предприятий. [2]

Канбан-это методика планирования задач, разработанная в сороковых годах. Суть канбан-доски заключается в наглядном расположении задач в соответствии с их статусом. Типичная доска делится на 3 колонки:

1. Задачи, которые необходимо выполнить (обычный to-do-лист).
2. Задачи, которые в текущий момент находятся в работе.
3. Задачи, которые уже выполнены и висят на доске исключительно для отслеживания прогресса.

Но доску можно дополнить и своими колонками. Например, в отдельный блок вынести реализованные функции, проходящие стадию проверки. Сценариев масса: можно приспособить канбан под что угодно, вплоть до семейного списка покупок на холодильнике. Скрам-доска– это канбан-доска для разработчиков, исповедующих Agile. Ее обычно дополняют колонками с заданиями на проверке и с отложенными делами.



Agile – это методика организации рабочего процесса, подразумевающая деление больших целей на мелкие, легко «перевариваемые» задачи и выполняющиеся в период спринтов (то есть недельных забегов активной работы).

Спринт создается на основе заранее сформированных целей. Цели же формируются исходя из пожеланий пользователей продукта. Лидер команды разработчиков организует спринт, добавляя туда те задачи, которые находятся у компании в приоритете и должны быть решены раньше остальных. Скрам-доски и продукты в духе Jira помогают организовывать спринты, следить за прогрессом команды и анализировать проведенную работу.

Команды разработчиков ПО работают лучше с помощью **Jira Software** — лучшего инструмента для agile-команд. Решение Jira Software разработано таким образом, чтобы каждый член вашей команды разработчиков мог планировать, отслеживать и выпускать превосходное ПО, даже мирового уровня.

**Jira Service Management** предоставляет обслуживание высочайшего уровня ИТ-командам, командам разработчиков, а также операционным и другим командам. Быстрая поставка ценности позволяет быстро создать службу поддержки и непрерывно адаптировать независимо от масштаба. Прозрачность рабочего процесса позволяет отслеживать всю деятельность предприятия благодаря открытой платформе для совместной работы. Путём связи задач в Jira можно получать массивы данных из других инструментов разработки ПО, чтобы расширить контекст для команд ИТ-поддержки и команд по эксплуатации и дать им возможность быстро реагировать на запросы, инциденты и изменения. Объединение команд по разработке и эксплуатации позволяет лучше удовлетворять потребности клиентов и управлять рисками, ускорять важнейшие участки разработки, уходить от монотонной работы и без труда развертывать изменения при поддержке полной регистрации каждого из них.

Бизнес-команды могут открыть возможности методики Agile и повысить эффективность совместной работы с помощью **Jira Work Management.**

Этот продукт разработан для команд по маркетингу, кадрам, финансам и других бизнес-команд. Представление по дням, неделям, месяцам, кварталам и т.д., отслеживание работы и связанных проектов, внесение изменений благодаря встроенному редактированию, наличие календаря, графика и настраиваемые доски для отслеживания работ и проектов вплоть до их завершения.

Кроме того, существует **Jira Align** — платформа для корпоративного agile-планирования и координации работы при любом масштабе. Шаблоны и решения, разработанные для каждой команды, а также Jira в качестве общего инструмента обеспечивают легкость и прозрачность работы в организации. Наглядное представление в режиме реального времени, согласование действий каждой команды со стратегией, оптимизация во благо клиентов. [2]

Программу Jira создавали для программистов. «Затачивали» каждый аспект под нужды разработчиков. Поэтому работает и выглядит она иначе. Не слишком универсально. В связи с этим имеется ряд конкретных сценариев, в которых применяется JIRA:

* Наглядная организация списка задач.
* Управление проектом и командой, занимающейся его развитием.
* Разработка ПО с нуля или добавление новых функций.
* Управление задачами, связанными с маркетинговой составляющей продукта.
* Отслеживание ошибок в программе и их своевременное исправление.

Вариантов применения Jira больше, но это основные. Они дают понять, кому вообще нужен подобный программный инструмент. [1]

1. **Алгоритм работы с Jira Software.**

Процесс работы с Jira можно разложить на 6 простых шагов:

Шаг 1. Создание проекта

1. Логотип Jira. В левом верхнем углу нажать значок главной страницы Jira.
2. В правом верхнем углу нажать Create project (Создать проект).

Шаг 2. Выбрать шаблон.

Библиотека шаблонов Jira содержит десятки шаблонов различных категорий и предназначена для быстрого и успешного начала работы команды. Можно выбрать шаблон для любых имеющихся продуктов Jira (Jira Software, Jira Service Management и Jira Work Management). На сегодняшний день Jira Software предлагает три следующих шаблона.

SCRUM



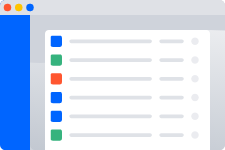
Для agile-команд, которые получают задачи из бэклога, планируют и оценивают работу в спринтах и регулярно поставляют результаты своей работы.

KANBAN



Для agile-команд, которые отслеживают ход работы как непрерывный процесс (без разделения на спринты) с упором на задачи, находящиеся в процессе выполнения. (Позволяет дополнительно использовать бэклог kanban.)

ОТСЛЕЖИВАНИЕ ОШИБОК



Для команд, которым не нужны доски и которые предпочитают работать с заданиями по разработке и багами как со списком.

**Типы проектов**

В шаблонах Scrum и Kanban предлагается выбрать тип проекта.

* **Проекты, управляемые командой**, подходят для независимых команд, желающих управлять собственными рабочими процессами и практиками в изолированном разделе.
* **Проекты компании** настраиваются и поддерживаются администраторами Jira. Этот тип проектов предназначен для команд, которые хотят стандартизировать способ работы среди множества команд, например организовать общий рабочий процесс.

Принципиальное различие между этими двумя типами проектов заключается в способе их администрирования и уровне управления (на уровне команды либо компании/администратора Jira).

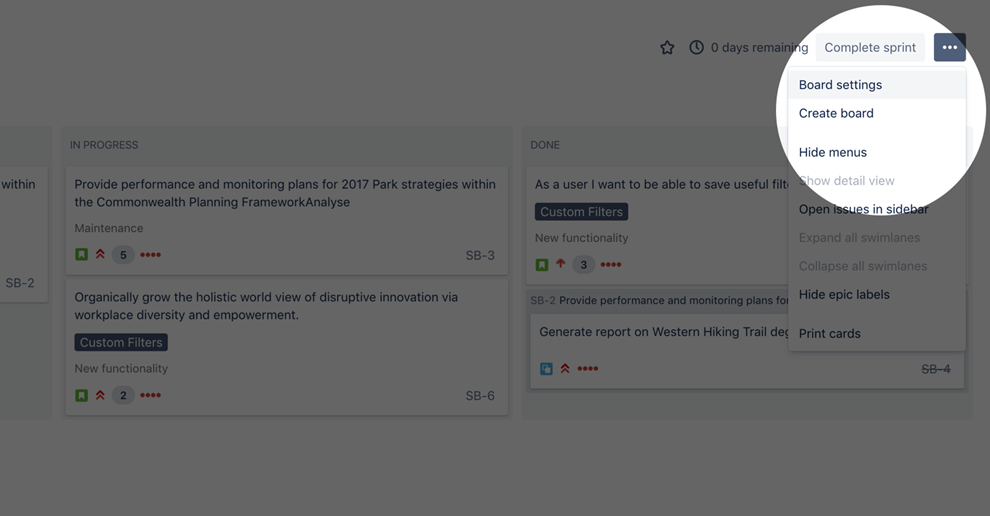
Шаг 3. Настройка столбцов.

Доска может содержать задачи из одного или нескольких проектов и наглядно демонстрирует рабочий процесс команды. Доски обеспечивают гибкое представление, что помогает управлять задачами и сообщает о незавершенной работе. Метод настройки столбцов на доске в шаблонах Scrum и Kanban зависит от типа проекта: в проекте, управляемом командой, настройка производится непосредственно с доски, а в проекте, управляемом компанией, — в настройках доски.

Хотя на доске многое можно настроить, на этом этапе рекомендуется только задать столбцы. На стадии подготовки нового проекта Jira Software важно, чтобы доска отражала то, как работает ваша команда.

Задать количество столбцов на доске можно следующим способом.

1. Перейти на доску команды, выбрав в меню проекта слева пункт Active sprints (Активные спринты) для scrum-проектов или Kanban board (Доска kanban) для kanban-проектов.
2. Нажать значок ••• > Board settings (Дополнительно > Настройки доски) в правом верхнем углу.

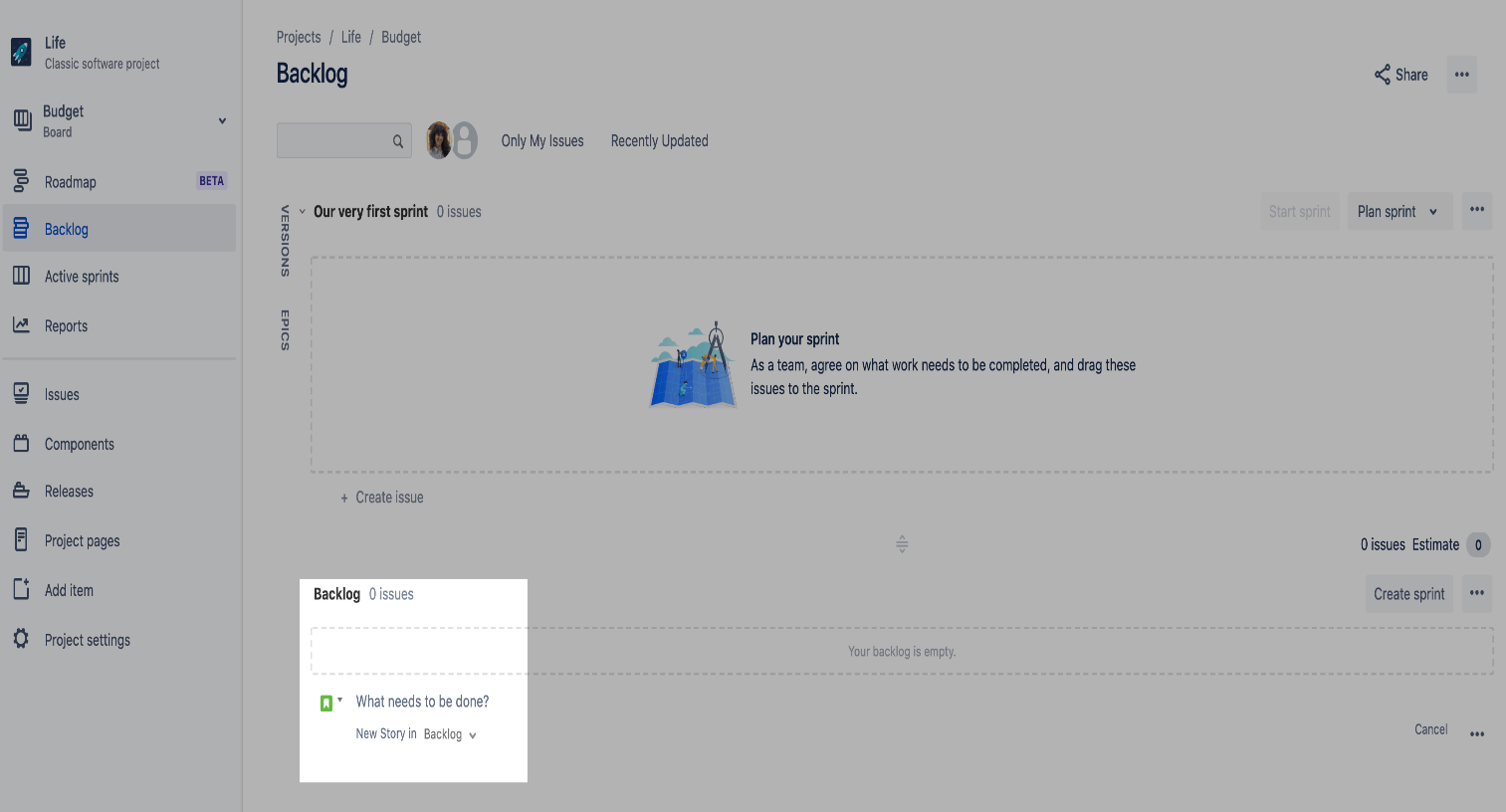


1. Нажать Columns (Столбцы).
2. Нажать Add column (Добавить столбец), чтобы добавить столбец для каждого этапа в рабочем процессе вашей команды.

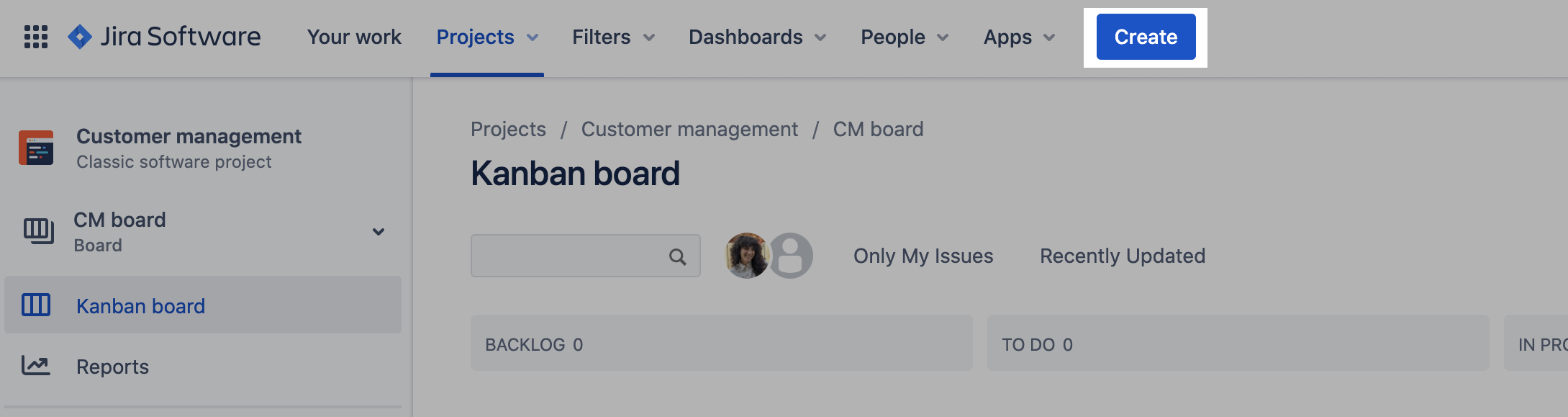
Например, если работу необходимо передать на этап контроля качества, прежде чем считать завершенной, можно добавить столбец под названием QA (Контроль качества) и поместить его перед столбцом Done (Завершено).

Шаг 4. Создать задачу.

Задачи — это составные элементы проекта Jira Software. Задача может отражать историю, эпик, баг, запланированную функцию или любое другое задание в рамках проекта. Scrum и kanban — это два подхода к управлению проектами по методике agile. Проекты scrum и kanban в Jira Software обладают разными возможностями, которые подходят для команд, работающих по той или иной методике. Если команда работает по методике scrum, нужно выбать Backlog (Бэклог) в меню проекта слева и нажать « + ». Создать задачу, чтобы пополнить список работ в бэклоге команды.

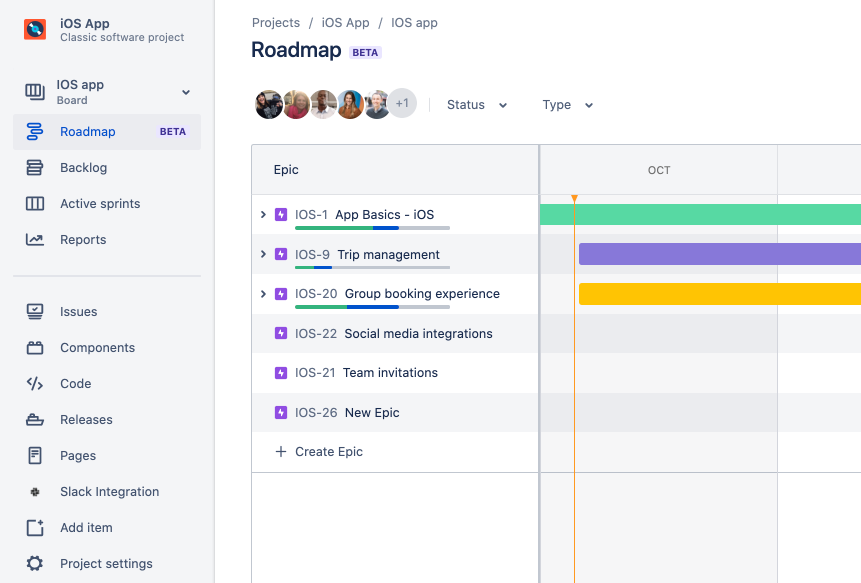


Если команда работает по методике Kanban, перейти на доску Kanban и выбрать Create (Создать) в основном меню. Задача появится в столбце Backlog (Бэклог).



Команды, использующие Scrum или Kanban, могут добавлять большие объемы работ в свои дорожные карты. Дорожная карта — это место для наглядного представления работы, ее планирования и управления ею.

* 1. В меню проекта выбрать пункт Roadmap (Дорожная карта).
  2. Начать вводить текст и нажать клавишу Enter, чтобы создать свой первый эпик.



После этого к задаче можно добавить небольшие подзадачи. Наведите курсор на эпик. Нажмите + Add a child issue (Добавить связанную задачу).

Настраивать задачи можно самыми разными способами.

Шаг 5. Пригласить участников команды.

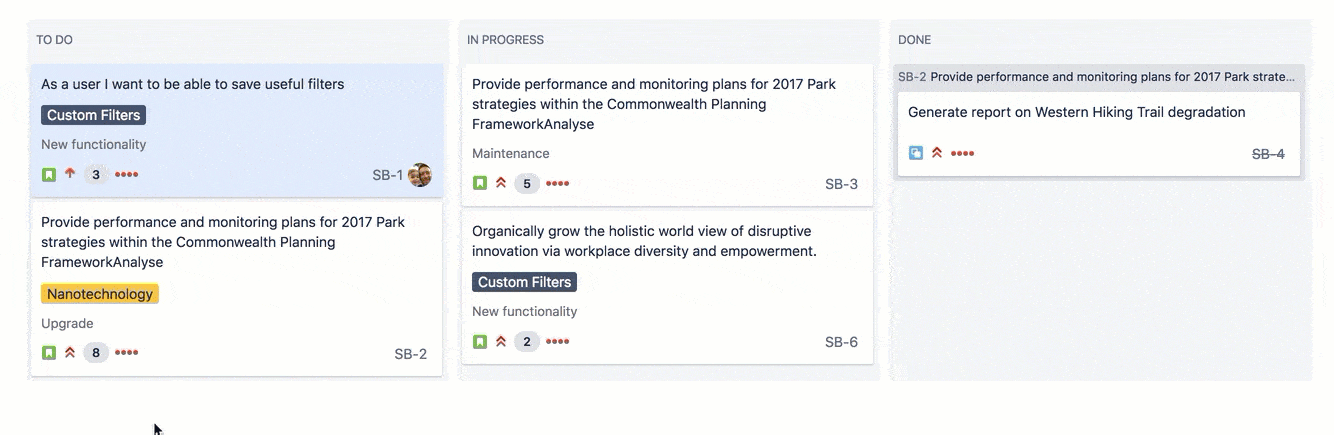
Когда на доске будет уже достаточно работы, можно начинать высылать приглашения участникам команды. Чтобы команда приступила к работе, нужно добавить каждого участника на сайт Jira следующим образом.

1. В меню проекта слева нажать Project settings (Настройки проекта).
2. Нажать People (Участники).
3. В правом верхнем углу нажать Add people (Добавить участников).
4. Найти адрес электронной почты участника команды и нажать Add (Добавить).

Шаг 6. Пусть работа движется.

Теперь, когда команда подключилась к сайту Jira, можно вместе работать и отслеживать процесс. Если работаете над Scrum-проектом, для отслеживания хода работы потребуется создать и [начать спринт](https://www.atlassian.com/ru/agile/tutorials/sprints). Если работаете над Kanban-проектом, можно следить за ходом работы на [доске](https://www.atlassian.com/ru/agile/tutorials/creating-your-agile-board). Чтобы следить за выполнением рабочих задач, нужно перемещать задачи из одного столбца в другой по мере прохождения ими рабочего процесса команды.

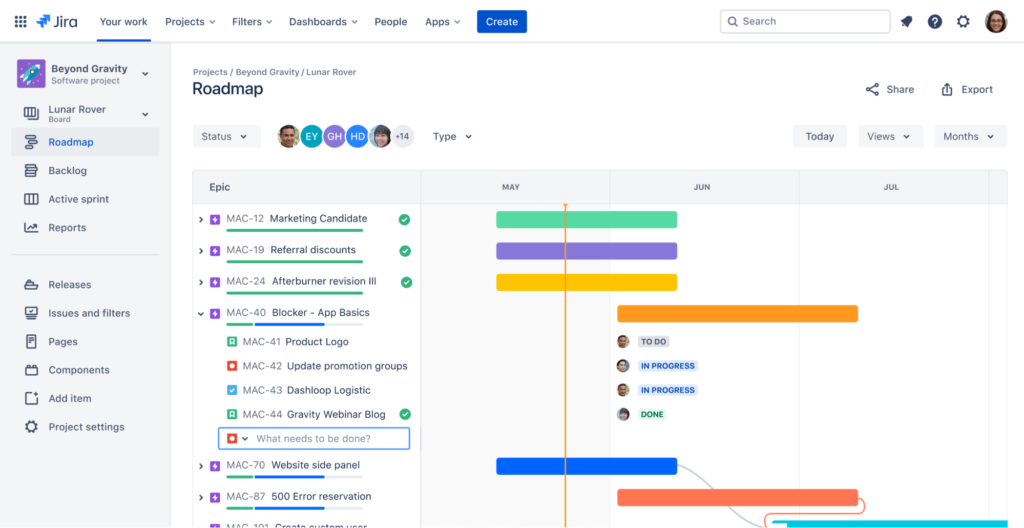
Можно комментировать задачи, чтобы все участники команды были в курсе происходящего. Это позволяет вести работу открыто, не ограничивая ее диалогом участников в переписке по электронной почте. [2]



* 1. **Как устроена Jira?**

Интерфейс.

Интерфейс Jira делится на несколько ключевых вкладок. Во вкладке «Projects» хранятся все канбан/скрам-доски, которые можно просматривать или редактировать. Фактически это основное рабочее пространство. Здесь же можно перейти в режим отслеживания релизов продукта, взглянуть на все активные спринты, проанализировать отчеты о проделанной работе и т.п. Также в списке вкладок есть окно с дашбордами – удобно скомпонованными аналитическими сводками. Отдельное окно со списком сотрудников, с которыми мы взаимодействуем, система планирования релизов на манер инструментов в духе OmniPlan и вкладка с приложениями от сторонних компаний, интегрированными в наш профиль Jira.



Задачи

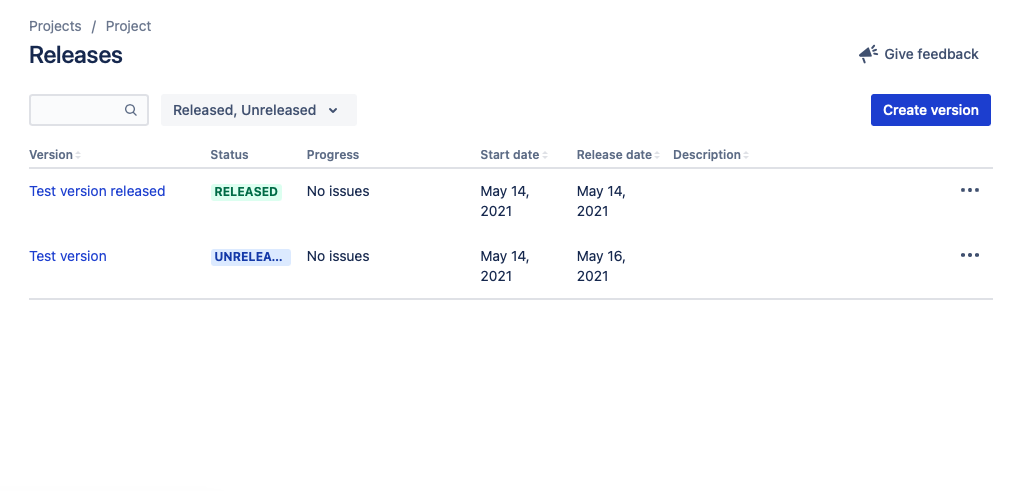
Задачи в оригинале называются Issues, что можно перевести как «проблемы». Issue – это единица информации. В нее закладывается либо какая-то функция, которую нужно реализовать, либо ошибка в программе, которую необходимо исправить. Issues – это составные части проекта и спринта. Именно список задач формирует рабочий процесс. Поэтому он и состоит из создания задач, наблюдения за ними, выполнения, анализа, дополнения, изменения и т.п. У задач в Jira есть типы. Для более удобной категоризации можно выбрать один из вариантов, например новую функцию, баг, подзадачу, эпики и т.п. Выбор типа задач зависит от целей команды и компании. Можно создавать свои типы для удобного распределения, фильтрации и поиска задач. Соответствующий раздел настроек находится в Project settings.

Дорожная карта (расписание).

В этом разделе можно создавать цели и планировать работу команды наперед. Ключевой единицей информации тут является эпик. Это объединение большого количества issues, связанных друг с другом. К примеру, если есть ряд новых функций для приложения, которые совместно формируют какую-то общую важную особенность ПО, то их объединяют в эпик как некую общую цель, к которой стремится команда в ходе спринта (или нескольких спринтов). На дорожной карте хорошо видны далеко идущие планы компании, визуально оформленные в своего рода горизонтальный календарь.

Релизы

Каждый набор новых функций в приложении или пакет исправленных ошибок отправляется к пользователям в виде новой версии этого самого приложения. Версионность – самый удобный, часто используемый и фактически ставший индустриальным способ развития программных продуктов. Поэтому в Jira такой акцент сделан на контроле новых версий. В соответствующем разделе можно создавать версии программ, указывать дату выпуска и закреплять за ними исправления багов, новые функции и issues, входящий в конкретный релиз. Здесь сразу видно, какая версия продукта должна выйти в ближайшем будущем, какие уже вышли и т.п. В общем, это еще один удобный способ планирования и отслеживания выполняемой работы.



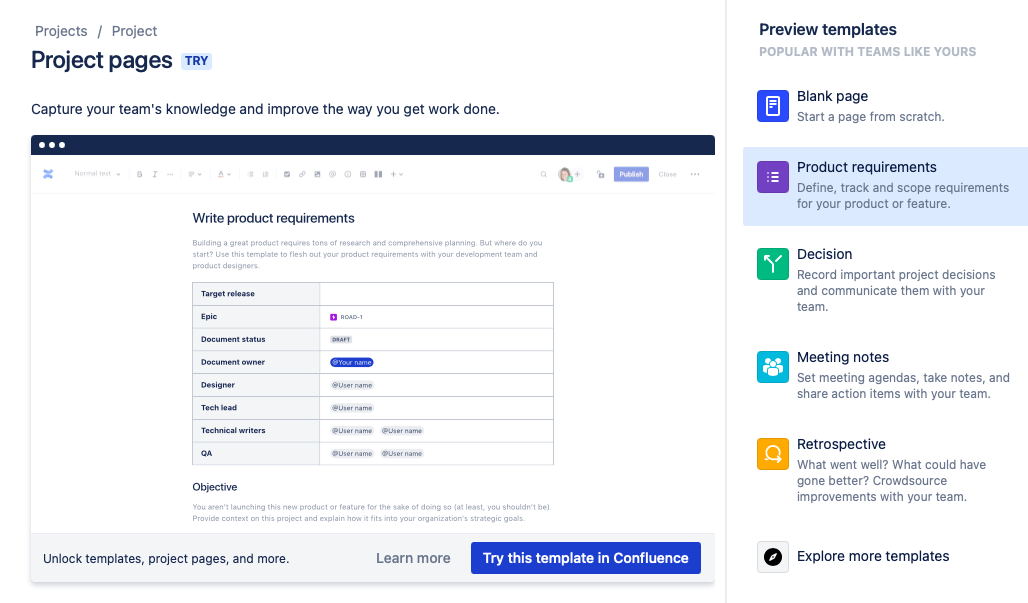
Код и деплой.

Одно из преимуществ Jira – возможность тесно интегрировать ее с другими продуктами, например с платформами Bitbucket, Github и Gitlab. Такое объединение добавляет дополнительный контекст в систему управления проектами. У лидеров команды появляется возможность наблюдать не только за прогрессом как за набором меняющихся активных задач, но и смотреть на реальные изменения в коде. Интеграция позволяет разработчикам напрямую отправлять каждый коммит в Jira, чтобы другие члены команды могли видеть изменения из условного Github прямо в системе управления проектами.

Pages

Еще одна разработка Atlassian – проект Confluence. Это что-то в духе Google Docs, только работающее в рамках Jira и менее функциональное. Это онлайн-текстовый редактор с базовыми инструментами для форматирования написанного. Суть Confluence в создании дополнительной удобной среды для общения лидеров команды и разработчиков. С помощью полноценного текстового редактора и неограниченного количества знаков гораздо проще изложить свои мысли и подробно рассказать о планах компании. В Pages по умолчанию есть несколько шаблонов для текста:

* Пустая страница с небольшим описанием функциональности Pages.
* Страница для описания продукта. Сюда можно вписать, как продукт выглядит в глаза клиента и что он в итоге должен собой представлять.
* Тестовая форма для указания глобальных целей компании с целью донести ее до команды и сподвигнуть к обсуждению дальнейших действий.
* Форма для заполнения расписания встреч и создания заметок по ходу общения с коллегами.
* Ретроспективная заметка с описанием проделанной работы. Здесь лидер команды указывает, что в ходе работы пошло хорошо, что пошло не очень и т.п.



Дашборды.

В Jira дашборды выводят кучу полезной информации. Всяческие отчеты, статистика комментариев, список выполняемых задач, аналитические данные, графики, таблицы, схемы. Все, что может быть полезно для аналитиков компании и бизнеса в целом. Дашборды позволяют собрать все необходимые данные в одно пространство без необходимости следить за процессом работы команды и фиксировать какие-то значимые аспекты, чтобы потом вручную делать аналитические сводки. В Jira можно формировать дашборды автоматически. Они подходят не только для аналитики, но и для постоянного наблюдения за тем, как протекает рабочий процесс, и для принятия радикальных решений в случае упадка производительности или возникновения других проблем.

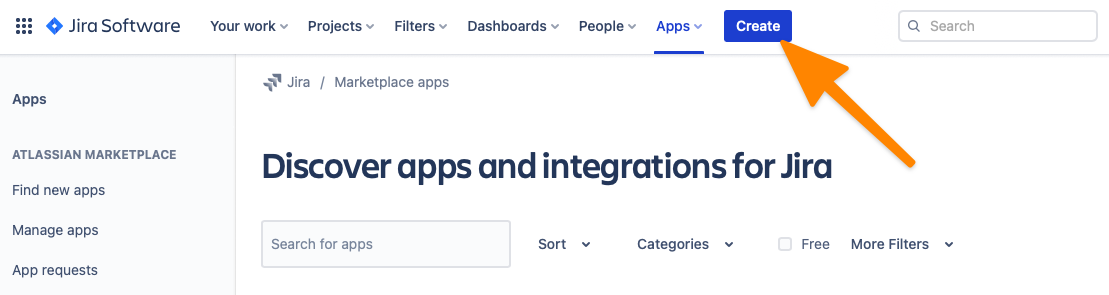
Плагины.

Jira можно сделать еще функциональнее, если подключить к ней плагины сторонних компаний. Некоторые из них продвигает сама Atlassian. В их числе интеграция с Git-системами, к примеру. Это один из наиболее распространенных и очевидных вариантов использования плагинов. Также разработчики Jira активно развивают идею тесного взаимодействия между пользователями системы управления проектами и пользователями мессенджера Slack. Есть даже отдельный набор программных модулей для интеграции одного в другое. В коллекции плагинов можно найти инструменты для создания диаграмм, более удобной визуальной презентации рабочего расписания, отправки задач по почте и т.п. Коллекция плагинов в Atlassian Marketplace насчитывает сотни дополнений. Как платных, так и бесплатных.

* 1. **Как создать задачу в Jira?**

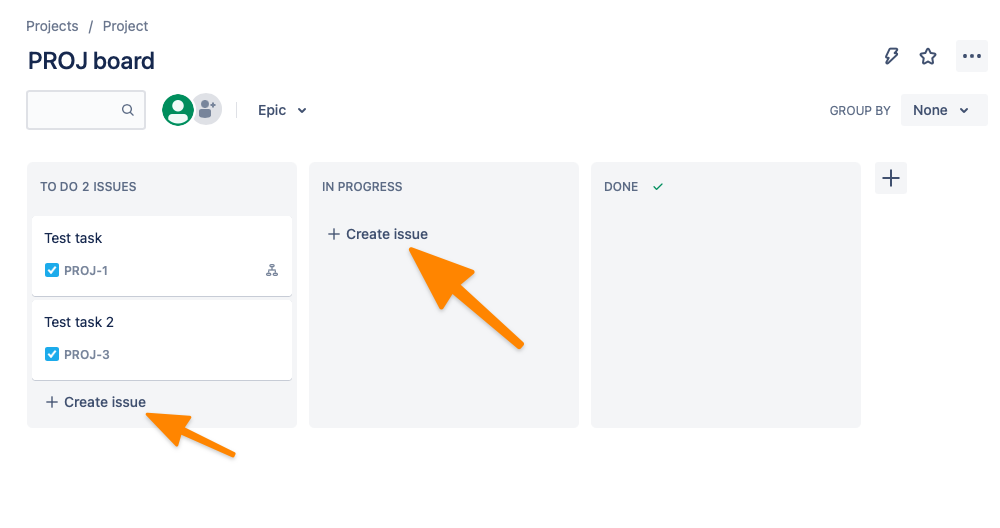
Создать новую задачу в Jira можно двумя путями:

* Кликнув по кнопке Create в верхней панели управления.



Нужно выбрать проект, в котором необходимо создать задачу.

* Кликнув по кнопке Create issue в нужной колонке канбан-доски.



Нужно указать название задачи и прописать дополнительные атрибуты. Уже после этого можно кликать по кнопке Create, и новая issue автоматически появится в списке и на выбранной доске. А если поставить галочку напротив Create another, то тут же появится окно для добавления еще одной задачи.

Список issues также можно импортировать из другого приложения. Для этого нужно загрузить в Jira CSV-файл с соответствующим содержанием.

Атрибуты задач.

Создавая issue, можно указать для нее ряд атрибутов:

* Summary. Краткое описание текущей задачи. Буквально в одно предложение.
* Description. Полноценное описание, если таковое требуется.
* Assignee. Член команды, которому нужно делегировать создаваемую задачу.
* Labels. Что-то вроде тегов для более удобной сортировки задач по другим признакам, не входящим в список типов.
* Fix version. К какой версии относится создаваемая issue.
* Story point estimate. Потенциальные трудозатраты, требующиеся на добавление новой функции или исправление бага.
* Reporter. Пользователь, который будет отчитываться за выполнение задачи.
* Attachment. Файл, прикрепленный к задаче. Это может быть что угодно: аудио, картинка, документ docx и т.п.
* Linked issues. То, с чем связаны создаваемые задачи (другие задачи/проекты).
  1. **Настройка отчетов в Jira.**

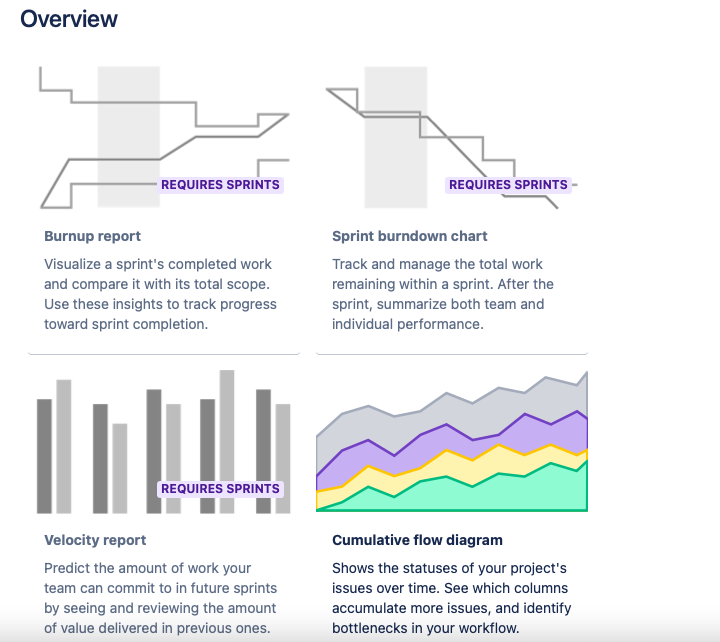
В графе Reports можно взглянуть на автоматически сгенерированные отчеты о проделанной работе. Пользователям Jira доступны 4 вида отчетов.

Burnup report -

График, показывающий результаты работы по конкретному спринту в сравнении с общей производительностью команды разработчиков. Его используют для оценки эффективности текущего спринта.

Sprint burndown chart -

График, показывающий, какой еще объем работы необходимо выполнить команде, чтобы продвинуться к завершению текущего спринта. Используется для оценки индивидуальной и общей производительности, а также для приблизительной оценки сроков реализации установленных планов.



Velocity report -

Используется для отображения потенциальной производительности команды в будущем. То есть на основе уже завершенных спринтов Jira пытается предугадать, как много задач удастся выполнить разработчикам в ходе следующего «забега».

Cumulative flow diagram -

Показывает, как менялся статус активных задач с течением времени. В каких колонках созданные issues задерживаются дольше всего. Используется для поиска так называемых бутылочных горлышек – проблемных этапов работы, на которых резко падает производительность всей команды.

Есть как минимум 5 способов сделать работу с Jira эффективнее.

Делите большие задачи на мелкие.

Комментируйте задачи.

Записывайте все выполненные действия.

Планируйте спринты.

Делайте записи на регулярной основе. [1]

* 1. **Создание баг-репорта.**

Баг – это всего-навсего дефект. Термин относится исключительно к проблемам в ПО. Баг-репорт (или тикет) – это письменное описание, детализирующее дефект. Как правило, баг-репорты пишутся в инструменте управления проектом вроде Jira. Баг и отчет о нем – это две разные вещи. Естественно, что не обнаруженные баги могут отлично существовать в продукте, когда отчетов о них еще не существует. Баг-репорт надо писать, когда обнаружена новая проблема, которая выглядит как дефект. Проблемы обнаруживаются в ходе такой тест-деятельности, как прогон автотестов или ручное исследовательское тестирование. Их также находят при разработке новых фич.

Вне зависимости от инструмента или процесса, хорошие баг-репорты содержат следующую информацию:

* *Краткое описание проблемы*
  + Краткое, однострочное описание дефекта
  + Которое ясно сообщает, что не так
  + И используется в заголовке репорта.
  + *Идентификатор репорта*
    - Уникальный идентификатор баг-репорта.
    - Как правило, генерируется автоматически инструментом управления вроде Jira
    - *Полное описание*
      * Подробное описание проблемы
      * Разъясняет всю релевантную информацию
      * Четкое и внятное
      * *Шаги воспроизведения*
        + Внятная процедура ручного воспроизведения проблемы
        + Могут быть шагами упавшего тест-кейса
        + Включают фактические и ожидаемые результаты
        + Случаи, когда дефект воспроизводится и НЕ воспроизводится
        + Версия продукта, ветка кода, название окружения, конфигурация системы, и так далее.
        + Воспроизводится ли дефект постоянно или "плавает"?
        + *Артефакты*

Приложите логи, скриншоты, файлы, ссылки, и т. п.

*Влияние*

Как влияет дефект на пользователя?

Блокирует ли он деятельность разработки?

Какие кейсы упадут из-за этого дефекта?

*Анализ первопричин*

Если это известно, объясните, чем вызван дефект

Если это неизвестно, предложите возможные причины

Внимание: явно отделяйте доказательства от спекуляций!

*Приоритезация*

* Если это возможно, назначьте задаче хозяина
* Назначьте ей серьезность или приоритет на основании гайдлайнов и здравого смысла
* Если это применимо, назначьте дедлайн
* И все остальное, что может понадобиться вашей команде.
* *Воспроизводимость* [3]

Баг- Репорт (Bug Report) — документ, содержащий отчет о любом недостатке в компоненте или системе, который может привести компонент или систему к невозможности выполнить требуемую функцию.

Атрибуты баг - репорта:

Основные -

* **ID** — уникальный идентификатор бага
* **Заголовок / Краткое описание / Тема / Summary / Title** — четко и кратко описывает суть бага. Оформляется в виде одного предложения, состоящего из трех частей отвечающих на вопросы “Что? Где? Когда?”. Редко бывает, что ответ на вопрос “Где?” или “Когда?” может опускаться, если он не дает полезной информации.
* **Шаги к воспроизведению** — **четкое, последовательное**описание шагов / действий, которые необходимо совершить, чтоб воспроизвести баг со всей необходимой информацией
* **Фактический результат** — результат, который мы **видим**
* **Ожидаемый результат —**результат, который мы **хотели / ожидали увидеть**
* **Серьезность —**показывает, насколько серьезные последствия от дефекта с точки зрения влияния на систему

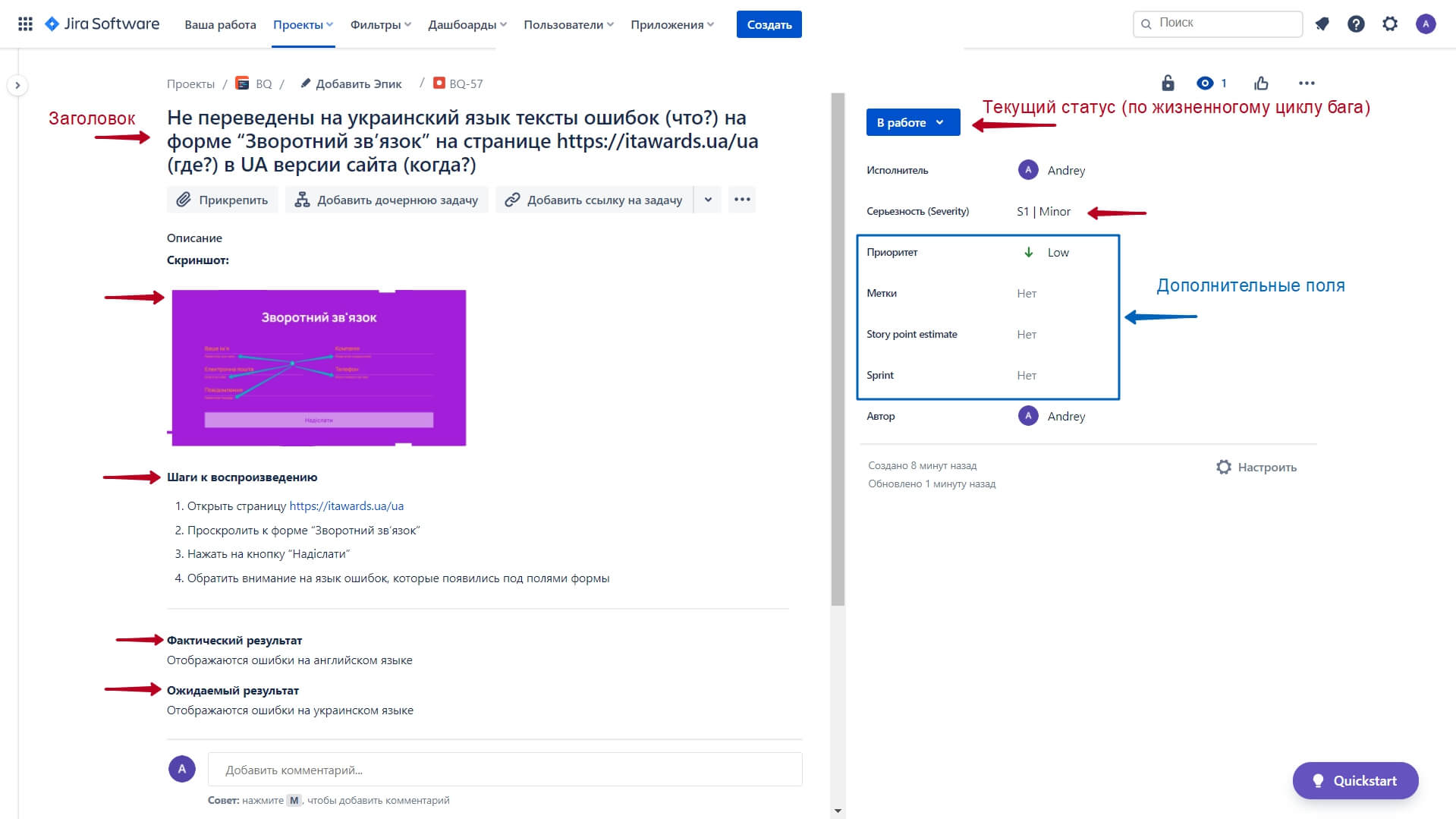
Дополнительные-

* **Скриншот / видео** — изображение / видео, которое четко и наглядно демонстрирует баг. Если видео или скриншот сделан качественно, его может быть достаточно для понимания сути ошибки и ее исправления
* **Требование —**ссылка на требование, которое не соблюдено. Наличие этой информации в 99% случаев предотвращает разговор “баг — не баг” и испорченное настроение
* **Тип бага —**для анализа “слабых” мест в ПО, баги могут разделять на типы
* **Приоритет —**очередь, в которой баг будет исправляться (Высокий -> Средний -> Низкий)
* **Дополнительные файлы —**файлы, которые нужны для воспроизведения бага (файлы определенного размера, типа, логи и т.п.)
* **Окружение —**информация об окружении, на котором воспроизводится баг (версия браузера, операционная система, размер экрана, тестовый сервер и т.п.)
* **Статус —**текущий статус бага в его жизненном цикле (Открыт, В работе…)
* **Автор —**человек, который создал баг (нужен для уточнения информации, если потребуется)
* **Исполнитель —**человек, которые работает над багом в данный момент времени
* **Комментарии —**обсуждение исправления ошибки
* **Версия —**версия ПО, в которой был обнаружен баг
* **Версия исправления —**версия ПО, в которую будет добавлено исправление бага

Алгоритм создания баг репорта.

1. Понять “суть” проблемы, а не ее проявление (если получится, но это требует технических знаний)
2. Воспроизвести дефект один-два раза (удостовериться, что он повторяется)
3. Проверить наличие найденного вами дефекта в системе управления дефектами (возможно, баг уже создали)
4. Написать заголовок (отвечает на вопросы “что? где? когда?”)
5. Написать основные поля отчета
6. Заполнить дополнительные поля отчета
7. Внимательно прочитать отчет. Убрать лишнее, добавить нужное!
8. **Еще раз перечитать отчет! (самый важный пункт)**
9. Сохранить отчет
10. Переназначить отчет либо проверяющему (если такой есть) либо разработчику (который будет исправлять ошибку)

Так выглядит баг-репорт в Jira:



Красным отмечены основные поля.

Синими отмечены дополнительные поля.

Основные поля являются обязательными для заполнения при создании бага, без них задача просто не сохранится.

**Пример отчёта о найденном баге:**

**Заголовок / Краткое описание / Тема**

**Скриншот / видео**

Скриншот / ссылка на скриншот

**Шаги к воспроизведению**

1. Открыть страницу https://
2. Проскролить (прокрутить)
3. Нажать на кнопку “Отправить”
4. Обратить внимание на язык ошибок, которые появились под полями формы

**Фактический результат**

Отображаются ошибки на английском языке

**Ожидаемый результат**

Отображаются ошибки на русском языке

**Серьезность**

S1 (minor)

Ошибки при создании баг- репорта.

1. Заголовок не отвечает на вопросы “Что? Где? Когда?”
2. Заголовок содержит лишнюю информацию (версии, окружения, учетные данные пользователей и т.п.)
3. Отсутствуют шаги для воспроизведения
4. Шаги для воспроизведения излишне детализированы
5. Отсутствует фактический и / или ожидаемый результат
6. Отсутствует ссылка на требование, которое проверялось (если такое есть)
7. Отсутствие скриншота / видеозаписи для UI/UX багов (сюда можно также добавить отсутствие выделения ошибки на скриншоте)
8. Грамматические ошибки / Техническая безграмотность. [4]

ВЫВОД: В ходе производственной практики мною были изучены следующие темы: Работа с Jira. Создание баг-репорта. Я узнал, что такое Jira? Для чего используют Jira. Изучил Алгоритм работы с Jira Software, как устроена Jira, как создать задачу в Jirа, как настраивать отчеты в Jira. Также я изучил, что такое баг и как создавать баг-репорты (отчёты), их атрибуты и алгоритм создания. Познакомился с ошибками при создании баг-репорта.

Список используемой литературы:

1. Статья: Как работать в Jira. Автор: Space Police. 20 мая 2020 г. Интернет-ресурс: <https://timeweb.com/ru/community/articles/kak-rabotat-v-jira>
2. Статья: Краткий обзор Jira. Интернет-ресурс: <https://www.atlassian.com/ru/software/jira/guides/getting-started/overview#jira-software-hosting-options>
3. Статья: Создание хороших баг-репортов. Автор: Энди Найт. 01.12.2020г. Интернет-ресурс:

<https://software-testing.ru/library/testing/bug-tracking/3469-writing-good-bug-reports>

1. Статья: Базовая теория. Баг и баг репорт. 21.06.2021г. Интернет-ресурс: BeQA.pro.

<https://beqa.pro/blog/%D0%B1%D0%B0%D0%B3-%D0%B8-%D0%B1%D0%B0%D0%B3-%D1%80%D0%B5%D0%BF%D0%BE%D1%80%D1%82/>